

Algoritmos y Estructuras de Datos II

Segundo parcial

Lunes 19 de octubre de 2020

Aclaraciones

- El parcial es individual y a libro abierto.
- Las consultas se responderán durante el día lunes 19 de octubre.
- El tiempo de entrega es **hasta el martes 20 de octubre a las 13:00hs.**
- **La entrega se realiza por GitLab.**
- Cada ejercicio se calificará con **Perfecto**, **Aprobado**, **Regular**, o **Insuficiente**.
- Para aprobar la cursada se deben tener todos los ejercicios aprobados de todos los parciales.
- Los ejercicios se recuperan por separado.

Ej. 1. Complejidad

Ej. 1.1. Sumas de consecutivos

Queremos determinar si un arreglo contiene una tira consecutiva de números que suman 0. Para ello tenemos las siguientes funciones:

```

función SUMARCONSEC(arreglo de enteros  $A$ , natural  $p$ , natural  $q$ )    ▷Precond.:  $p + q \leq \text{LONG}(A)$ 
|
|    $s := 0$ 
|   para  $i := p \dots p + q - 1$  hacer
|       |    $s := s + A[i]$ 
|   fin
|   devolver  $s$ 
fin

```

```

función CONSECSUMAN0?(arreglo de enteros  $A$ )
|
|    $n := \text{LONG}(A)$ 
|   para  $\text{cuantos} := 1 \dots n$  hacer
|       |   para  $\text{pos} := 0 \dots n - \text{cuantos}$  hacer
|           |   si  $\text{SUMARCONSEC}(A, \text{pos}, \text{cuantos}) = 0$  entonces
|               |   devolver true
|           |   fin
|       |   fin
|   fin
|   devolver false
fin

```

Como precondición de SUMARCONSEC pedimos que $p + q \leq \text{LONG}(A)$.

Supongamos que los arreglos se pasan por referencia y que las operaciones $\text{LONG}(\cdot)$ y $\cdot[\cdot]$ se calculan en tiempo $O(1)$.

1. Dar una función f tal que la complejidad temporal de $\text{SUMARCONSEC}(A, p, q)$ sea $\Theta(f(q))$. Justificar.
2. ¿Cuál es la complejidad temporal del mejor caso de $\text{CONSECSUMAN0?}(A)$? Justificar.
3. Dar una función f tal que la complejidad temporal de $\text{CONSECSUMAN0?}(A)$ en el peor caso sea $O(f(\text{LONG}(A)))$. Justificar.

Ayuda: dependiendo del camino que tomen, puede ser útil saber que $\sum_{i=1}^n i^2 = \frac{1}{6}n(n+1)(2n+1)$.

Ej. 1.2. Funciones suaves

Sea \mathbb{Z}^+ el conjunto de los enteros positivos, es decir $\{1, 2, 3, \dots\}$.

Una función $f: \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ es *eventualmente no decreciente* si existe un $N \in \mathbb{Z}^+$ tal que $f(n) \leq f(n+1)$ para todo $n \geq N$. (Notar que esto implica que $f(x) \leq f(y)$ para todo $N \leq x \leq y$.)

Dados una función $f: \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ y un $b \in \mathbb{Z}^+$, notamos $f_b: \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ a la función definida por $f_b(n) = f(bn)$ para todo $n \in \mathbb{Z}^+$.

Decimos que una función f es *b-suave* si f es eventualmente no decreciente y $f_b \in O(f)$. Decimos que una función es *suave* si es *b-suave* para todo $b \in \mathbb{Z}^+$.

1. Probar que si una función f es *b-suave* entonces $f_b \in \Theta(f)$.
2. Probar que si una función f es *b-suave* entonces f es *a-suave* para todo $a < b$.
3. Probar que si una función f es *b-suave* entonces f es b^k -suave para todo $k \in \mathbb{Z}^+$.
Pista: notar que, para todo $x \in \mathbb{Z}^+$, $b^x n = b(b^{x-1}n)$ y con esto probar que $f_{b^x} \in O(f_{b^{x-1}})$.
4. Sea f una función *b-suave* para algún $b \in \mathbb{Z}^+$ mayor o igual a 2. Probar que f es suave.
Pista: usar los dos ítems anteriores.
5. Determinar cuáles de las siguientes funciones son suaves:
 - $f(n) = n^k$, con $k \in \mathbb{Z}^+$
 - $g(n) = n \log n$
 - $h(n) = n^n$

Pista: notar que, gracias al ejercicio anterior, alcanza con ver cuáles son 2-suaves.

Ej. 2. Invariante de representación y función de abstracción

La gerencia de Caballito Futbol quiere que especifiquemos un sistema para administrar sus canchas. Desde el comienzo contamos con un conjunto de canchas fijas, cada una del mismo tamaño, en la que pueden jugar 10 jugadores (ni más ni menos). El complejo además cuenta con una sala de espera, donde esperarán pacientemente quienes deseen jugar.

Durante el día van llegando grupos al predio, cada grupo con una cantidad arbitraria de personas. **Siempre que haya al menos 10 personas esperando y alguna cancha libre, esta se asignará automáticamente.** Sin embargo, estas 10 personas no necesariamente serán del mismo grupo, sino que serán escogidas por el sistema. Esto quiere decir que el algoritmo con el que se arman los equipos se definirá en la etapa de diseño. Por último, cada vez que se libere una cancha, si hay suficientes personas, se asignará una cancha de la misma forma que describimos previamente.

Nos pidieron especificar con un TAD el Sistema de Caballito Futbol (SCF) descripto, teniendo en cuenta que además se desea saber cuál fue la persona que más asistió a cada una de las canchas (de haber más de una, elegir alguna sin ninguna preferencia puntual). Para ello, Alexis propuso la siguiente solución, en la cual **las asistencias se actualizan cuando la gente se retira de la cancha**:

TAD SCF**observadores básicos**

Canchas : SCF \rightarrow conj(Cancha)

Ocupadas : SCF \rightarrow conj(Cancha)

Jugando : SCF $s \times$ Cancha $c \rightarrow$ conj(Persona) $\{c \in \text{Canchas}(s) \wedge_L c \in \text{Ocupadas}(s)\}$

SalaEspera : SCF \rightarrow conj(Persona)

Asistencias : SCF $s \times$ Cancha $c \rightarrow$ Dicc(Persona, nat) $\{c \in \text{Canchas}(s)\}$

generadores

$$\text{IniciarSCF} : \text{conj}(\text{Cancha}) \ c \longrightarrow \text{SCF} \quad \{\#c > 0\}$$

$$\text{LlegaGrupo} : \text{SCF} \ s \times \text{conj}(\text{Persona}) \ c \longrightarrow \text{SCF} \quad \{\#c > 0 \wedge c \cap \text{EnElPredio}(s) = \emptyset\}$$

$$\text{LiberarCancha} : \text{SCF} \ s \times \text{Cancha} \ c \longrightarrow \text{SCF} \quad \{c \in \text{Canchas}(s) \wedge_L c \in \text{Ocupadas}(s)\}$$
otras operaciones

$$\text{MaxJugador} : \text{SCF} \times \text{Cancha} \longrightarrow \text{Persona} \quad \{\#\text{claves}(\text{Asistencias}(s, c)) > 0\}$$

$$\text{EnElPredio} : \text{SCF} \longrightarrow \text{conj}(\text{Persona})$$

...

axiomas

...

Fin TAD

Ahora queremos representarlo mediante la siguiente estructura:

scf **se representa con** estr, donde

estr es tupla

$$\langle \begin{array}{ll} \text{esperando} & : \text{secu}(\text{Persona}) \\ \text{enCancha} & : \text{dicc}(\text{Persona}, \text{Cancha}) \\ \text{libres} & : \text{conj}(\text{Cancha}) \\ \text{asistencias} & : \text{dicc}(\text{Cancha}, \text{dicc}(\text{Persona}, \text{nat})) \end{array} \rangle$$

Informalmente, esta representación cumple las siguientes propiedades:

- En *esperando* están todas las personas que se encuentran en la sala de espera, en algún orden.
- Para toda persona que se encuentra jugando en el predio, *enCancha* nos dice en qué cancha está jugando.
- En *libres* están todas las canchas libres.
- En *asistencias* guardamos para cada cancha del predio cuántas veces jugó cada persona que la visitó alguna vez. En caso de que nadie haya jugado en esa cancha todavía, el diccionario asociado a ella es vacío.

Se pide:

1. Escribir (formalmente y en castellano) el invariante de representación.
2. Escribir la función de abstracción.